De la sémantique des langages de programmation à la vérification sémantique des sites Web

Thierry Despeyroux
Inria, France
E-mail: thierry.despeyroux@inria.fr
http://www-rocq.inria.fr/~tdespeyr/td-fr.html

Brigitte Trousse
Inria, France
E-mail: brigitte.trousse@inria.fr
http://www-sop.inria.fr/axis/AXIShtml/personnel/Brigitte.Trousse/bri-eng.html

Résumé:

La quantité d'information accessible sur le Web est phénoménale et la recherche d'une information pertinente et cohérente devient une gageure. Le but principal du Web Sémantique est de faciliter et de mécaniser la recherche d'information en formalisant une matière jusque là plutôt textuelle. Notre démarche est différente puisque nous nous intéressons à la construction des sites et voulons aider au contrôle et au maintient de la cohérence sémantique de ces derniers en utilisant des techniques utilisées habituellement pour définir la sémantique formelle des langages de programmation.

Introduction

Les sites Web sont devenus des produits de consommation courante et se pose maintenant la question de la qualité de ces sites. Ceci est d'autant plus important que les sites Web ont une vie complexe: en général plusieurs auteurs, des parties qui peuvent être générées (nous ne nous intéressons pas dans un premier temps aux sites dynamiques), des mises à jour et des refontes fréquentes. Le risque est donc grand de voir s'introduire, même dans un site initialement bien conçu, des erreurs ou des incohérences.

Jusqu'à présent, les efforts entrepris dans le domaine du Web sont allés essentiellement dans deux directions:

• La structuration syntaxique, avec XML, les DTD, XML-shema, les style sheets et XSLT [17]: il s'agit d'isoler le contenu des pages de la présentation qui en est faite et de définir une syntaxe abstraite afin de contraindre la structure de l'information; au delà des travaux de définition menés pas le W3C, il faut citer des outils de manipulation de documents XML, qui peuvent prendre en compte la légalité des documents produits vis-à-vis des DTD [8, 10].

• L'annotation sémantique, à l'aide d'ontologies utilisant RDF, RDF-Schema [17] ou DAML+OIL [16], l'idée étant de présenter une représentation utilisable par programme du contenu de certaines pages du Web et d'annoter le contenu de ces pages [14, 6, 5] ou bien de s'appuyer sur une ontologie pour vérifier le contenu d'un site [15].

Outre l'annotation de page Web, la même information peut aussi exister sous des formes multiples, en particuler pour des raisons d'accessibilité. Nous nous trouvons donc maintenant avec une redondance d'information que ce soit sous la forme de données ou de métadonnées. Manipuler plusieurs représentations d'une même information n'est pas un problème en soit mais encore faut-il pouvoir s'assurer de la cohérence des différentes représentations.

Dans ce papier, nous présentons une vision très "génie logiciel" du contrôle de la cohérence dans un

site Web. L'idée générale est d'utiliser une méthode de spécification formelle utilisée dans le monde de la sémantique des langages de programmation pour définir une sémantique pour un site Web, ou plus largement pour un système d'information, afin de générer un outil de vérification.

1 De la vérification des programmes à celle des sites Web

Une grosse application (ou programme) se présente le plus souvent comme un ensemble de fichiers et de librairies. Un programme comme "make" [12] permet de gérer les dépendances entre ces fichiers et de déclencher de façon minimale l'appel d'un compilateur. Outre le fait que ce dernier permet de générer un code objet, il permet surtout de vérifier la légalité du programme vis-à-vis de la sémantique statique du langage de programmation. Ainsi il n'est pas rare de trouver des parties déclaratives, dans lesquelles des objets, des types ou des procédures sont décrits pour être utilisés ailleurs. Il faut donc vérifier que les utilisations sont conformes aux déclarations.

La particularité de ce genre de contraintes est qu'elles ne sont pas locales: elles peuvent lier des occurrences distantes dans un même fichier ou même dans des fichiers différents. Deuxième particularité, ces contraintes ne sont pas indépendantes du contexte: un "environnement" qui permet de connaître les objets visibles à un endroit particulier doit donc être disponible pour permettre les vérifications. Ces contraintes ne peuvent donc pas être décrites de façon syntaxique.

Un site Web (que nous définissons simplement comme un ensemble de pages) possède de grandes similitude avec un programme, et en particulier leur représentation sous forme de termes (ou arbres), mais aussi beaucoup de différences:

- un très grand nombre de fichiers,
- une grande dissémination de l'information, avec en particulier un recours systématique aux références "en avant",
- un langage d'expression peu formel, avec des parties utilisant des langues naturelles,
- la possibilité d'utiliser du multimédia, par exemple des images,
- des contraintes sémantiques qui ne sont pas données par un langage de programmation, mais qui doivent être explicitées par le concepteur du site,
- la nécessité de faire appel à des ressources externes pour exprimer la sémantique (par exemples des thésaurus, des ontologies ou des programmes d'analyse d'images).

Les métadonnées, qui sont maintenant de façon standard présentées à l'aide d'une syntaxe XML, même si d'autres syntaxes concrètes sont utilisées (par exemple la notation N3 [1] dans le cas de RDF), entrent aussi dans ce schéma ainsi que les données des bases de données ou les réponses aux requêtes à ces mêmes bases de données. Nous pouvons donc dire que l'émergence de XML nous a donné un cadre permettant de traiter un système d'information a priori hétérogène de façon uniforme.

Notre but est donc de proposer des méthodes et des outils qui puissent, par utilisation d'une commande telle que "make", vérifier l'intégrité sémantique d'un système d'information et fournir des diagnostiques précis si ce n'est pas le cas.

2 Exemples de contraintes sémantiques

Pour mieux appréhender la notion de contrainte sémantique, nous passons en revue deux exemples typiques.

- Un annuaire thématique est un site dans lequel des documents ou des sites externes sont classés pour en faciliter la recherche. Une équipe éditoriale est en charge de produire une classification arborescente par thèmes et sous-thèmes. La sémantique implicite de ce genre de classification est qu'un sous-thème "a du sens" dans le contexte des thèmes englobants, et cette sémantique doit bien sûr être maintenue au cas où le site est modifié.

L'exemple qui suit illustre ce propos:

Catégorie: Loisirs

Sous-Catégories: Sport, Voyages, Chirurgie, Musique, Cinéma

Cette sémantique fait appel au sens des mots d'une langue naturelle. Pour vérifier ce genre de contrainte sémantique, il va falloir utiliser des ressources externes qui peuvent être des thésaurus ou des ontologies préexistantes ou construites au fur et à mesure de la construction et de la vie du site, peu importe vraiment. Ce qui compte ici est

de pouvoir mécaniser la "relecture" du site.

• Un site institutionnel a la particularité de présenter une organisation: organigramme, explication du mode de fonctionnement, description des services, directions, etc. L'information est donnée avec un certaine redondance, avec bien sur les risques de manque de cohérence inhérents. Comme dans un programme, il faut alors identifier une partie qui fait foi: l'organigramme ou un annuaire, supposé maintenu à jour et qui permet ensuite de vérifier que ce qui est dit ailleurs est exacte, en particulier lors de modifications d'affectation de personnes ou de restructurations de services. Cette "partie qui fait foi" peut revêtir la forme d'un document assez formel comme une ontologie, faire appel à une base de données, mais est très souvent un simple document XML.

Le problème de cohérence entre données et métadonnées ou entre données redondantes se rencontre à de multiples endroits. Ce peut être de vérifier qu'une légende se rapporte bien à une image: dans ce cas, on peut vouloir confronter à l'aide d'outils linguistiques la légende avec des annotations décrivant l'image ou faire appel à des outils d'analyse de l'image; ça peut être de vérifier la cohérence entre diverses versions d'un site, par exemple pour traiter les problèmes d'accessibilité; ça peut être aussi de vérifier une cohérence entre une requête à une base de données et le résultat de cette requête pour détecter une erreur de formulation de requête et vérifier la plausibilité de la réponse; dans le cas de pages annotées, par exemple en RDF, ce peut être de vérifier que cette description est toujours correcte après modification de la page.

3 Vers une formalisation sémantique des sites Web

Comme nous l'avons vu plus haut, les sites Web (ou d'autres systèmes d'information) tout comme les programmes peuvent être représentés par des termes typés. Les méthodes formelles utilisées pour les langages de programmation semblent donc a priori adaptées. Nous avons utilisé pour nos expériences la sémantique naturelle [2, 9] qui est une sémantique opérationnelle dérivée de la sémantique opérationnelle de Plotkin[11] et inspirée par les séquents de Gentzen [13]. Elle est exécutable par compilation en Prolog afin de pouvoir générer des vérificateurs ou des compilateurs et permet de construire des preuves.

Dans ce système, la sémantique est exprimée sous formes de règles d'inférence et d'axiomes. Ces règles expriment comment démontrer des propriétés du point courant, appelé sujet, en fonction de ses sous-composants et à partir d'un environnement (ou ensemble d'hypothèses).

À titre d'illustration, la règle suivante explique comment la partie "instructions" d'un programme dépend de la partie "déclarations":

$$\frac{\emptyset \vdash Decls \rightarrow \rho \qquad \rho \vdash Insts}{\vdash \textbf{declare } Decls \textbf{ in } Insts}$$

Les déclarations sont analysées dans un environnement initial vide $\emptyset$ pour construire un environnement ρ qui est utilisé pour vérifier la partie "instructions".

Ces règles d'inférence peuvent être lues dans deux sens: si on a prouvé la partie haute on en déduit la partie basse; mais aussi dans un mode plus opérationnel: pour montrer la partie basse, il faut d'abord montrer la partie haute.

La règle suivante est un axiome qui exprime le fait que le type d'une variable est celui que l'on peut trouver dans l'environnement.

$$\rho \vdash \textbf{var } X : T \qquad \{X : T\} \in \rho$$

L' "exécution" de la sémantique consiste à construire un arbre de preuve par "empilement" des règles de sémantique auparavant instanciée par la donnée.

Les expériences que nous avons menées jusqu'à présent [3, 4] montrent que ce style de sémantique convient parfaitement, mais est trop lourd à mettre en place dans le cas des systèmes d'information. En effet, comme on peut le voir sur cette illustration, la récursion dans un tel système est explicite et il faut donc au moins une règle par construction syntaxique (i.e. par tag dans le cas de XML). D'autre part, la manipulation de l'environnement peut être fastidieuse, en particulier en raison du problème de déclaration en avant.

Nous nous orientons donc vers un langage permettant de générer les règles de sémantique naturelle ou tout au moins de générer un code qui reste dans l'esprit de la sémantique naturelle, avec comme objectif la simplicité:

• pas de récursion explicite,

• ne parler que des points d'intérêt,

• une gestion simple de l'environnement,

• permettre la combinaison de règles (gestion de point de vue),

• gérer automatiquement la question des déclarations en avant.

Pour l'instant nous avons programmé directement ce qui pourrait être le code généré en Prolog, et en sommes dans la phase de design d'un langage. Le choix de Prolog viens surtout du fait que c'est le langage dans lequel est compilée la sémantique naturelle. En fait Prolog est particulièrement bien adapté à notre problème, puisqu'il manipule des termes (et les pages XML sont des termes) ainsi que des prédicats.

Dans ce qui suit, nous présentons brièvement les principaux traits de ce futur langage.

• Les schémas, ou motifs, permettent de désigner et repérer des occurrences dans des arbres XML. Pour cela XML est étendu avec la notion de variable logique. Dans les exemples qui suivent, le nom des variables logiques commence par le signe `$`.

• Un environnement local permet de stocker des valeurs. L'application de certaines règles peut être conditionnée par cet environnement local. Nous avons structuré cet environnement sous la forme de variables que l'ont peut lire (`=`) ou affecter (`:=`).

• Un environnement global permet de stocker des prédicats, autrement dit des assertions qui sont déduites lors de l'application d'un règle. La syntaxe utilisée pour l'instant est celle de Prolog, mis à part le fait que les variables logiques sont repérées par le signe `$`. Le signe `=>` indique qu'un prédicat doit être ajouté à l'environnement global.

• Des tests qui permettent de générer des messages d'erreur lorsque certaines conditions ne sont pas remplies, ces prédicats de test étant pour l'instant eux aussi directement écrit en Prolog. L'expression `? préd / erreur` signifie que si le prédicat `préd` est faux, alors il faut générer le message d'erreur.

Les règles de sémantique comportent deux parties: la première exprime dans quelles conditions la règle peut être utilisée à l'aide d'un schéma et de test sur l'environnement local; la deuxième permet de décrire une action de modification de l'environnement local, d'ajouter une assertion à l'environnement global ou de générer un test.

La récursion est implicite, ainsi que la propagation des environnements et des messages d'erreur. Ne reste donc par rapport aux règles habituelles de la sémantique naturelle que des actions de modification et de test des environnement.

L'exemple qui suit permet d'illustrer ce qui précède. Nous sommes dans un organigramme qui présente l'organisation des services d'une entreprise. Cette page est donc considérée comme faisant foi vis-à-vis du reste du site.

```
<service>
 <nomservice>$X</nomservice>
 $_
</service>   =>   service:=$X
```

La règle ci-dessus indique que dans tous les sous-arbres la variable d'environnement `service` vaudra la valeur unifiée avec la variable `$X`. Cette valeur pourra donc être retrouvée par la suite, comme dans la règle qui suit. `$Y` représente ici le "reste" du contenu du tag "service".

Contrairement à XSL où il est possible de remonter chercher une donnée présente dans l'arbre entre sa racine et le point courant, nous devons stocker les valeurs utiles dans l'environnement local. En revanche, ces données peuvent être calculée, ce qui n'est pas le cas avec XSL.

```
<resp>$P</resp>   et
 service=$X   =>   resp($P,$X)
```

Nous sommes dans le contexte du service `$X` et `$P` est le responsable du service. `resp($P,$Y)` est une assertion qui est donc disponible pour l'ensemble des pages du site. Dans la mesure ou une ontologie du domaine traité existe, nous pouvons très bien imaginer y faire référence; en effet les triplets de RDF peuvent être vus comme des cas particuliers des prédicats de Prolog.

```
<agent>$P</agent>
  ? affectation($P,$X)
    / erreur(sansaffectation,$P)
```

Cette dernière règle illustre la génération de messages d'erreur: dans un texte, l'utilisation de la balise `agent` permet de s'assurer que celui-ci a toujours une affectation. Le code généré par les règles de sémantique est suffisamment riche pour permettre une identification précise de la localistion de l'erreur dans le source.

Conclusion

Dans cet article nous avons montré comment des techniques importées d'un autre domaine (la sémantique des langages de programmation) pouvait être appliquées au Web. Notre démarche est complémentaire d'autres travaux de recherches parfois proche: ainsi dans [7], des méthodes de programmation logique sont utilisées pour spécifier et vérifier des contraintes d'intégrité sur la structure des sites Web. Pour notre part nous nous intéressons au contenu sémantique. Notre objectif est non seulement de construire des outils de spécification mais aussi de pouvoir générer des outils de vérification faciles à utiliser.

References

[1] T. Berners-Lee. Ideas about web architecture - yet another notation notation 3. W3C http://www.w3.org/DesignIssues/Notation3.html.

[2] T. Despeyroux. Executable Specification of Static Semantics. In *Semantics of Data Types, Lecture Notes in Computer Science,Vol. 173*, June 1987.

[3] T. Despeyroux and B. Trousse. Semantic verification of web sites using natural semantics. In *RIAO 2000, 6th Conference on "Content-Based Multimedia Information Access", College de France, Paris, France*, April 2000.

[4] T. Despeyroux and B. Trousse. Maintaining semantic constraints in web sites. In *AACE WebNet 2001 Conference, Orlando, Florida*, October 2001.

[5] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, R. Studer, and A. Witt. On2broker: Lessons Learned from Applying AI to the Web. Technical report, Institute AIFB, 1998b.

[6] D. Fensel, R. Decker, M. Erdman, and R. Studer. Ontobroker : the Very High Idea. In *Proceedings of the 11th Internation FLAIRS Conference (FLAIRS-98)*, May 1998.

[7] M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. Verifying integrity constraints on web sites. In *IJCAI*, pages 614–619, 1999.

[8] H. Hosoya and B. Pierce. Xduce: A types xml processing language. In *Proceedings of Third International Workshop on the Web and Databases*, May 2000.

[9] G. Kahn. Natural Semantics. In *Proceedings of the Symp. on Theorical Aspects of Computer Science, TACS*, Passau, Germany, 1987. LNCS 247, Springer-Verlag, Berlin. also Inria Research Report 601, February 1987.

[10] E. Meijer and M. Shields. XMλ: A functional programming language for constructing and manipulating xml document, 1999. Draft, http://www.cse.ogi.edu/ mbs/pub/xmlambda/.

[11] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.

[12] R. M. Stallman and R. McGrath. *GNU Make: A Program for Directing Recompilation, for Version 3.79*. Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA, Tel: (617) 876-3296, USA, 2000.

[13] E. Szabo. *The Collected Papers of Gerhard Gentzen*. North-Holland, Amsterdam, 1969.

[14] F. van Harmelen and D. Fensel. Practical Knowledge Representation for the Web. In D. Fensel, editor, *Proceedings of the IJCAI'99 Workshop on Intelligent Information Integration*, 1999.

[15] F. van Harmelen and J. van der Meer. Webmaster: Knowledge-based Verification of Webpages. In *Twelfth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE'99*, 1999.

[16] W3C. Daml+oil (march 2001) reference description. W3C http://www.w3.org/TR/daml+oil-reference.

[17] W3C. Xml, xsl, xml schema and rdf recommendations or submissions. W3C http://www.w3.org/.